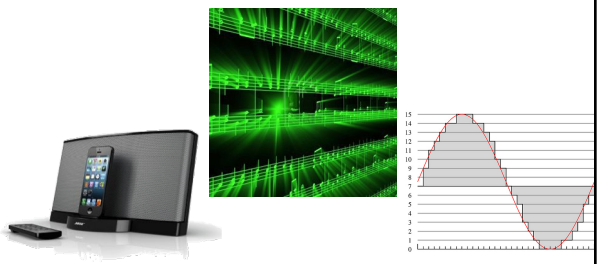
Penn Engineering ESE



Lecture #9 – Operating Systems (OS)

ESE 150 – Digital Audio Basics

ESE150 Spring 2019

1

ESE150 Spring 2019

Preclass 1

- What things can your phone do while you are listening to an MP3?

2

ESE150 Spring 2019

Observation

- We want our devices (including our phones) to do many things at once.

3

ESE150 Spring 2019

Multiple Tasks

- We could…
  - Dedicate a separate processor for every task we want to perform
- How many would we need?
- Maybe
  - Need dozen processors for our Phone

4

ESE150 Spring 2019

But….

- MP3 Play
  - 44,000 samples per second decoded
  - 500 cycles to decode a sample
  - How many instructions per second require?
- What fraction of a 10^9 instruction per second processor does this use?

5

ESE150 Spring 2019

Observation

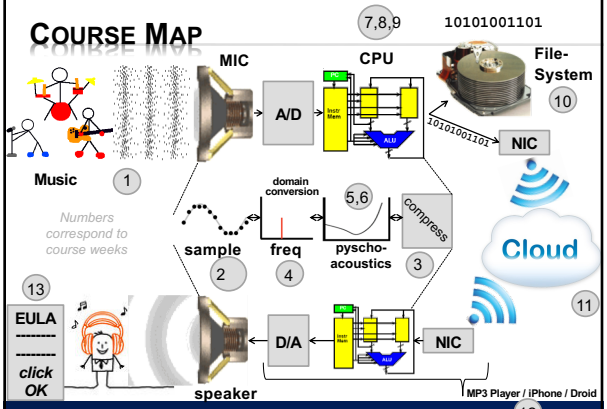
- If we dedicate a processor to MP3 decoding
  - It will sit idle most of the time
- MP3 decoding (and many other things) do not consume a modern processor
- Idea: Maybe we can share the processor among tasks?

6

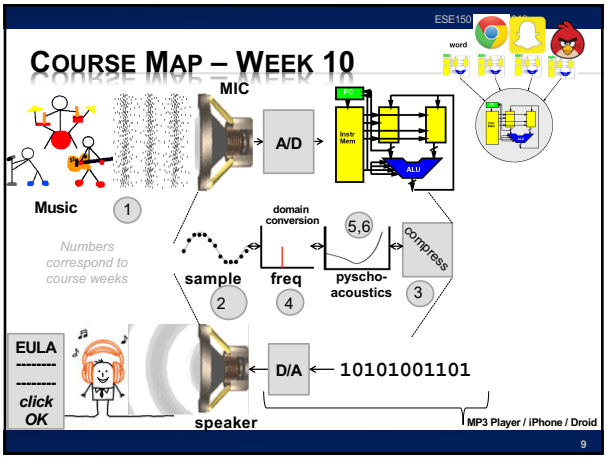
OUTLINE

- Setup Need / Opportunity
- Where are we
- Role of Operating System
- Virtualization

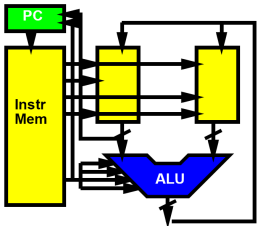
COURSE MAP



COURSE MAP – WEEK 10



"STORED-PROGRAM" PROCESSOR

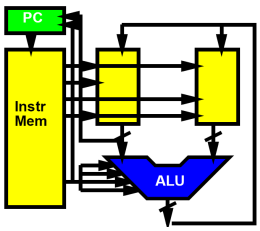


- By filling in memory, can program to perform any computation

ROLE OF OPERATING SYSTEM

PROGRAMMING THE PROCESSOR

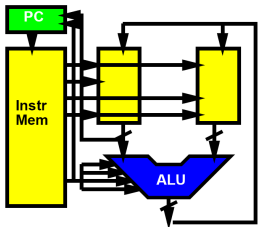
- Think about: How do we change the program in the memory



- What if had to reboot machine… for every application?
  - Change flash card
  - Download over USB
    - …that is what we have to do for the Arduino…

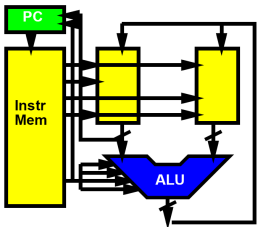
More than one Program?

- How could we have multiple applications?
  - (just run one at a time for now)



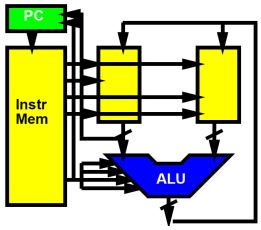
Multiple Running Programs?

- How can multiple applications share this processor?



Coordination?

- Does every program need to know about every other program?
  - Implications and Viability?
- IoT device that runs 3 unchanging tasks?
- Smart phone that allows 3rd party apps
  - Think: AppStore, Google Play
- Your laptop



Role of Operating System

- Higher-level, shared support for all programs
  - Could put it in program, but most programs need it!
  - Needs to be abstracted from program
- Resource sharing
  - Processor, memory, "devices" (net, printer, audio)
- Polite sharing
  - Isolation and protection
  - *Fences make Good Neighbors* – R. Frost
- Idea: Expensive/limited resources can be shared in time – OS manages this sharing

Virtualization

Virtualization

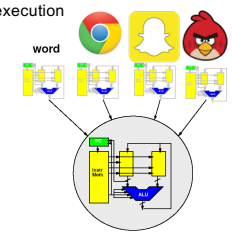
- Providing an abstract view separate from the physical view
- Hides physical view
- Provides abstract view to software
  - Abstract from physical resource limits

IDEA

- **Virtualize the processor**
  - Make it look like we have multiple processors
  - With each program running on its own processor
- **"Own" processor**
  - Can put data in memory where it wants
  - Doesn't have to worry about another program scribbling over memory
  - Its state is preserved and isolated
  - Looks like it runs all the time on the processor
    - Doesn't need to be programmed to allow other programs to run

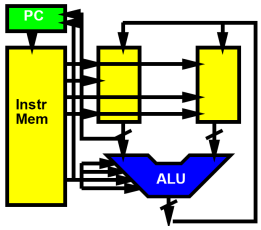
TERMINOLOGY: PROCESS

- **Process**
  - A virtualization of the physical processor
    - an instance of a program in execution
  - Virtual processor



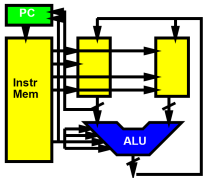
WHAT DOES OUR PROGRAM SEE?

- **Physically**
  - One processor
    - One PC
    - One data memory
    - One instruction memory
  - These are its **state**
    - Terminology: context



EXECUTING THE PROGRAM

- **To execute program**
  - Keep track of state of machine
    - Value of counter (Program counter)
    - Contents of instruction memory
    - Contents of data memory



EXECUTION EXERCISE

- **We're going to simulate the computer and watch the processor state**
- **See worksheet**

EXECUTION: GETTING STARTED

		DMEM			
Cycle	PC	0	1	2	3
Initial	0	5	35	255	66
+1	1	5	35	0	66
+2	2	5	1	0	66

A

IMEM	0	DMEM[2]=DMEM[2]-DMEM[2]; PC=PC+1
	1	DMEM[1]=DMEM[2]+1; PC=PC+1

Execution Exercise

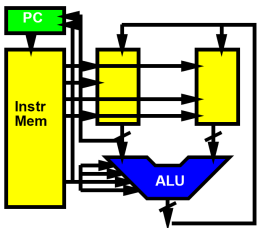
- **Simulate one of the 2 cases (as indicated on your worksheet) for the 12 cycles shown.**
- **Work in pairs**
  - …check each other…

Execution Exercise

- **What is the state for the +12 cycle? (A, B)**
- **What is the state for the +6 cycle? (A, B)**

One Processor, One Program

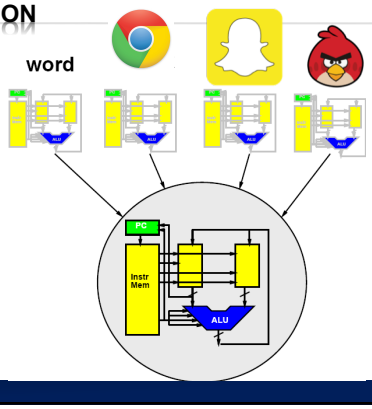
- **On the physical machine, can only run one program**
  - Why?
    - One PC
    - One memory



Virtualization

- **Make it look like we have multiple resources**
  - Multiple processors
- **Provide abstraction of large* number of processors**
  - Each program gets its own processor
    - Each program gets its own machine state
  - * "large" enough to approximate infinite

Virtualization

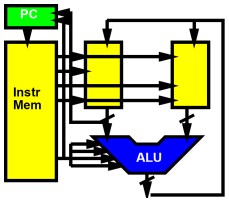


Key Idea

- **Can capture state of a processor**
  - All the information that defines the current point in the computation

Executing the Program

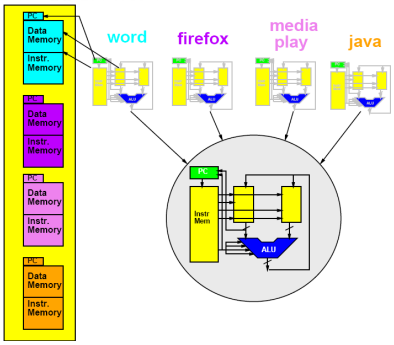
- **State of the processor**
  - Value of Program Counter (PC)
  - Contents of instruction memory
  - Contents of data memory



Key Idea

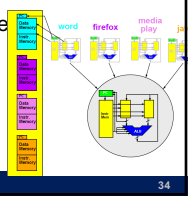
- **Can capture state of a processor**
  - All the information that defines the current point in the computation
  - i.e. program counter, data and instruction memory
- **Can save that in memory**
  - A different memory from what the process sees
  - (could be different range of addresses)
- **Fully represents the running program**
- **Can restore that from memory to the processor**
- **Can save/restore without affecting the functional behavior of the program**

State in Memory



Sharing Processor

- **Now that we can save/restore the state**
- **Can share processor among processes**
  - (Restore state; run for time; save state)
- **Isolation: none of the processes need to know about each other**
  - Each thinks it has the a whole machine
  - Just need to restore/save state around epochs where the process gets to run on the processor

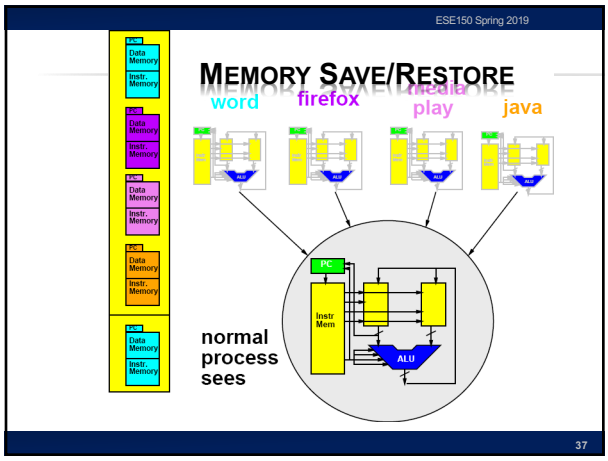


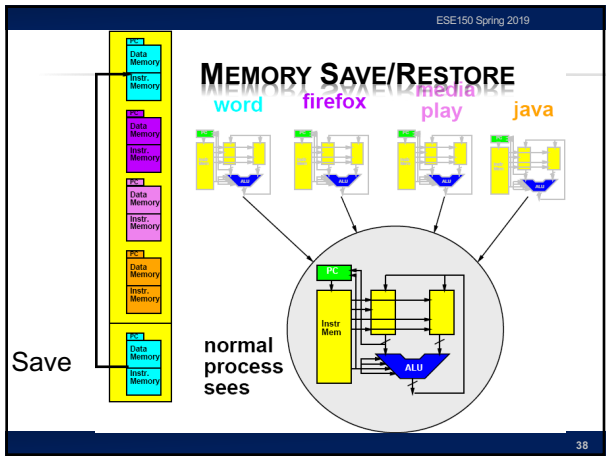
Memory?

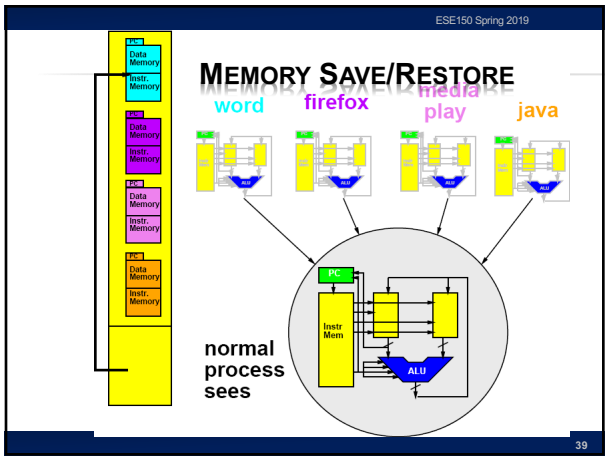
- **"save all of memory" ?**
  - Must have more memory
  - Enough to hold all the memory of all the running programs == all the processes
- **Each program has view that it owns machine**
  - Each may put program in same place?
  - Shouldn't have to know about other programs, where they use memory

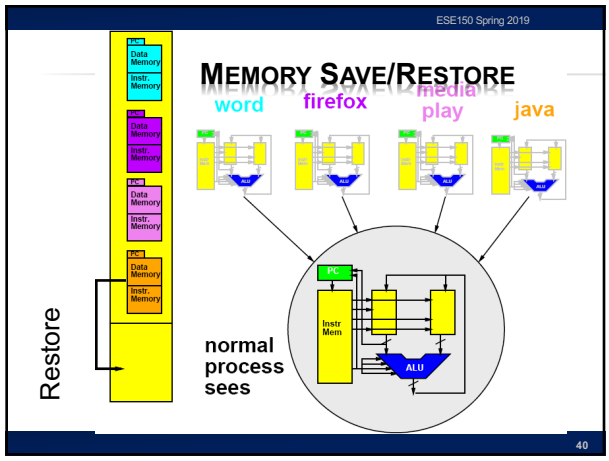
Saving Memory?

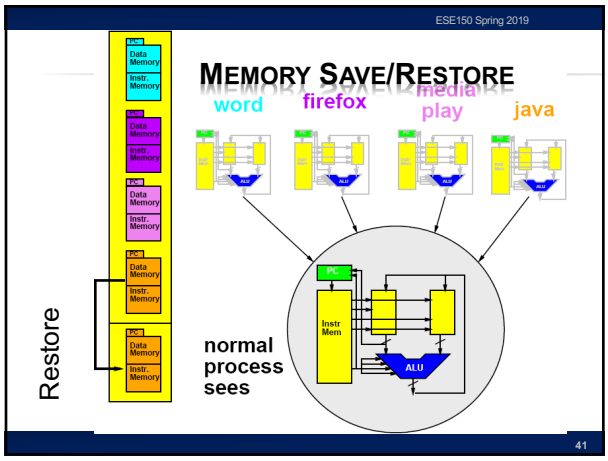
- **Each program has view that it owns machine**
  - Each may put program in same place?
  - Shouldn't have to know about other programs, where their stacks are…
- **Could:**
  - Have programs operate 0…max_process_mem
  - Copy data in and out of this range
  - Keep elsewhere
    - more memory not visible to program
    - On disk

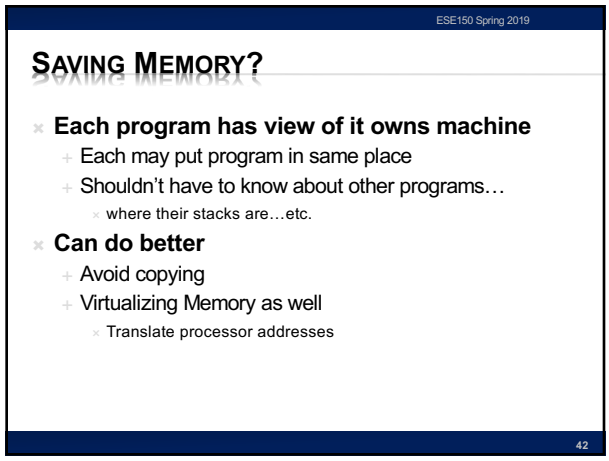

ESE150 Spring 2019
MEMORY SAVE/RESTORE
word
firefox
media play
java
normal process sees
37


ESE150 Spring 2019
MEMORY SAVE/RESTORE
word
firefox
media play
java
Save
normal process sees
38


ESE150 Spring 2019
MEMORY SAVE/RESTORE
word
firefox
media play
java
normal process sees
39

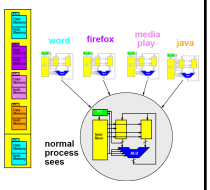

ESE150 Spring 2019
MEMORY SAVE/RESTORE
word
firefox
media play
java
Restore
normal process sees
40


ESE150 Spring 2019
MEMORY SAVE/RESTORE
word
firefox
media play
java
Restore
normal process sees
41


ESE150 Spring 2019
SAVING MEMORY?
Each program has view of it owns machine
Each may put program in same place
Shouldn't have to know about other programs…
where their stacks are…etc.
Can do better
Avoid copying
Virtualizing Memory as well
Translate processor addresses
42

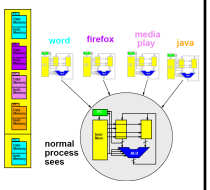
Management Program

- **Need another program → process**
  - Manage swap of running processes
  - Decide what to run next
  - Decide when to stop a process
- **…process manager/scheduler**



Time-Sliced Sharing

- **Simplest version:**
  - Run each process for 10,000 cycles
  - Then swap to next process
  - Looks like each of n process runs on a processor 1/n-th the speed of the real processor
- **More sophisticated:**
  - Assign uneven time to processes
  - Also change when process…
    - waits for input
  - What are cases where
    - Uneven time appropriate?
    - Valuable to switch on input?



Time Switch Exercise Demo

- **Simulating a case:**
- **Processor runs A for 6 cycles**
  - Then stores off to memory.
- **Processor runs B for 6 cycles**
  - Then stores off to memory
- **Processor reads A state from memory and runs for another 6 cycles**
- **Processor reads B state from memory and runs for another 6 cycles**
- **What should happen? (results should we get?)**

Time Switch Exercise

- **Write down the +6 cycle state from the opposite case**
  - This is your “swap back in” of task

Time Switch Exercise

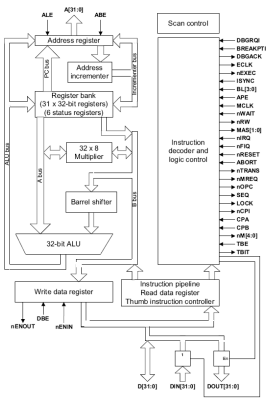
- **Simulate from +6 cycles**
- **What is the state for the +12 cycle?**
- **Compare earlier solutions**

Review: Key Idea

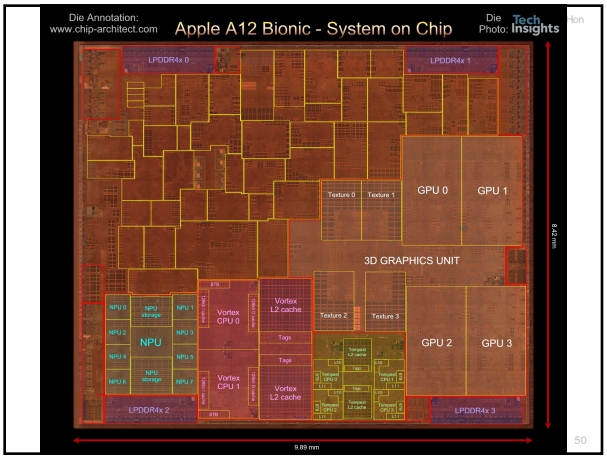
- **Can capture state of a processor**
  - All the information that defines the current point in the computation
  - i.e. program counter, data and instruction memory…
- **Can save that in memory**
  - A different memory from what the process sees
  - (could be different range of addresses)
- **Fully represents the running program**
- **Can restore that from memory to the processor**
- **Can save/restore without affecting the functional behavior of the program**

iPod Processor

- **Early based on PortalPlayer series**
  - Two ARM7TDMI cores
  - 80MHz each
- **Guesses are ARM7 or ARM8**



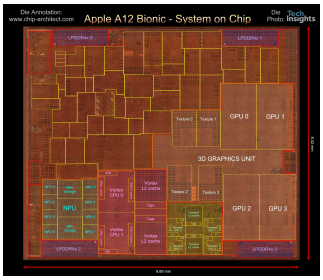
49



50

Apple A12

- **84mm², 7nm**
- **7 Billion Tr.**
- **iPhone XS**
- **6 ARM cores**
  - 2 fast
  - 4 low energy
- **4 custom GPUs**
- **Neural Engine**
  - 5 Trillion ops/s?



51

Upcoming Lab

- **Explore linux and processes on linux**
  - See processes sharing processors
  - Lab out

52

Big Ideas

- **Virtualize hardware**
  - Identify state; save/restore from memory
- **Program view: owns complete machine**
- **Allows programs to share limited physical hardware (e.g. processor)**
  - Provide illusion of unlimited hardware
- **Operating System is the program that manages this sharing**

53

Learn More

- **CIS380 – Operating Systems**

54